Amendment to the Claims:

1. (Currently Amended) A method, comprising:
   executing a first Operating System (OS) generated thread generated by an OS, the OS generated thread being associated with a first system state;
   encountering a non-privileged, user-level programming instruction;
   creating in hardware without intervention of the OS, responsive to the non-privileged level programming instruction, a first shared resource thread (shred) that is associated with a first private portion of a first application state, wherein the first shred shares a shared portion of the first application state and the first system state with at least a second shred; and
   executing, responsive to the non-privileged, user-level programming instruction, the first shred concurrently with at least the second shred; and
   communicating between the first shred and the second shred via one or more shared registers that are adapted to be updateable by the first and the second shred, wherein the one or more shared registers are private to shreds that share the shared portion of the first application state and the first system state.

2. (Previously Presented) The method of claim 1, wherein the first private portion of the first application state is associated with at least one of a plurality of registers including general purpose registers, floating point registers, MMX registers, segment registers, a flags register, an instruction pointer, control and status registers, SSE registers, and a MXCSR register.

3. (Cancelled)

4. (Previously Presented) The method of claim 3, further comprising:
   creating a second operating system generated thread to execute a second process in response to a privileged non-user level programming instruction, the second

operating system generated thread to be associated with a second system state; and

creating a third shred, in response to encountering a second non-privileged user-level programming instruction associated with the second process, wherein the third shred is to be associated with a shared portion and a private portion of a second application state,

wherein the second operating system generated thread and the third shred do not share a portion of the first application state.

5. (Previously Presented) The method of claim 1, wherein the first shred and the second shred share a current privilege level and share a common address translation.

6. (Previously Presented) The method of claim 1, further comprising: receiving a non-privileged user-level programming instruction that encodes a shred.

7.–9. (Cancelled)

10. (Previously Presented) The method of claim 1, further comprising: scheduling, responsive to a user-level programming instruction, the first shred and the second shred without intervention of the operating system.

11. (Previously Presented) The method of claim 1, wherein said communicating is performed via a user-level shred signaling instruction.

12. (Previously Presented) The method of claim 11, further comprising: storing at least the first private portion of the first application state in a memory responsive to receipt of a context switch request.

13. (Previously Presented) The method of claim 1, further comprising: handling with user-level exception handler code an exception generated during execution of the first shred, without intervention of the operating system.

14. (Previously Presented) The method of claim 13, further comprising: receiving the exception from an application program; and determining whether to report the exception to the operating system.

15. (Previously Presented) The method of claim 1, wherein the shred is to perform input/output (I/O) operations.

16. (Previously Presented) The method of claim 1, wherein the first and second shreds are to perform computation functions.

17. (Currently Amended) An apparatus, comprising:
execution resources to execute a first shared resource thread ("shred") concurrently with at least a second shred in response to receiving a first user instruction; and
a shared register to be directly accessible by a second user instruction from the first and the second shred to provide communication between the first shred and the second shred, wherein the shared register is not accessible shreds that do not share an application state with the first and the second shred.

18.–19. (Cancelled)

20. (Previously Presented) The apparatus of claim 17, wherein the shared register comprises a first register to enable an operating system or BIOS to enable multithreading architecture extensions for user-level multithreading.

21. (Previously Presented) The apparatus of claim 17, wherein the shared register comprises a first register to be atomically updated to synchronize data between the first and the second shred.

22. (Previously Presented) The apparatus of claim 17, wherein the execution resources include one or more processor cores capable of executing multiple shreds concurrently.

23. (Previously Presented) The apparatus of claim 17, wherein the shared register is to hold at least a portion of a state shared among the first shred and the second shred.

24. (Previously Presented) The apparatus of claim 17, wherein the first shred and the second shred share a current privilege level and share a common address translation.

25. (Previously Presented) The apparatus of claim 17, further comprising logic to execute a user-level instruction to create the first shred.

26. (Previously Presented) The apparatus of claim 17, wherein the shared register comprises a first register to be utilized as a register semaphore to synchronize data between the first and the second shred.

27. (Previously Presented) The apparatus of claim 17, further comprising a group of registers to share a system state among the first shred and the second shred.

28. (Previously Presented) The apparatus of claim 17, further comprising a group of registers, which does not include the shared register, to hold a private portion of a state to be associated with the first shred.

29.–31. (Cancelled)

32. (Previously Presented) The apparatus of claim 17, further comprising: a user-level exception mechanism to report an exception to the first shred.

33. (Previously Presented) The apparatus of claim 17, further comprising: an exception mechanism to report an exception to an operating system.

34. (Previously Presented) The apparatus of claim 32, further comprising:
a mechanism to detect a first exception associated with the first shred and a second exception associated with the second shred;
wherein the exception mechanism includes a prioritizer to prioritize the first and the second exceptions;
and wherein the exception mechanism is further to report only one of the prioritized first and second exceptions at a time to the operating system.

35. (Currently Amended) A[[n article of manufacture]] <u>non-transitory computer readable medium</u> including data that, when accessed by a machine, cause the machine to perform operations comprising,
receiving user-level programming instructions to execute a plurality of user-level threads of execution, wherein each of the plurality of user-level threads of execution are to be associated with a private state of an operating system (OS)-generated thread and to share a system state with the OS-generated thread;
creating, in hardware without intervention of an OS, the plurality of user-level threads of execution to be executed concurrently on multiple instruction sequencers of a processor in the machine in response to receiving the user-level programming instructions; and
communicating between the plurality of user-level threads of execution via one or more shared registers that are to be directly updateable by the plurality of user-level threads<u>, wherein the one or more shared registers art to not be accessible by shreds that do not share the system state with the OS-generated thread</u>.

36. (Currently Amended) The computer readable medium ~~article of manufacture~~ of claim 35, wherein the private state is associated with at least one of a plurality of registers including general purpose registers, floating point registers, MMX registers, segment registers, a flags register, an instruction pointer, control and status registers, SSE registers, and a MXCSR register.

37. (Currently Amended) The computer readable medium ~~article of manufacture~~ of claim 35, wherein the private state includes a private portion of an application state, and wherein the plurality of user-level threads of execution are also to share a shared portion of the application state, the shared portion of the application state to be associated with at least one of a plurality of registers including a control register, a flags register, memory management registers, a local descriptor table register, a task register, debug registers, model specific registers, shared registers, and shred control registers.

38. (Currently Amended) The computer readable medium ~~article of manufacture~~ of claim 35, wherein the computer readable medium ~~machine-accessible medium~~ further includes data that causes the machine to perform operations comprising: sharing a state among the plurality of user-level threads of execution; and storing the state in one or more registers.

39. (Currently Amended) The computer readable medium ~~article of manufacture~~ of claim 35, wherein the plurality of user-level threads of execution share a current privilege level and share a common address translation.

40. (Currently Amended) The computer readable medium ~~article of manufacture~~ of claim 35, wherein the user-level programming instructions include an instruction to create one or more of the plurality of user-level threads of execution.

41. (Currently Amended) The computer readable medium ~~article of manufacture~~ of claim 35, wherein the computer readable medium ~~machine-accessible medium~~ further includes data

that causes the machine to perform operations comprising communicating among the plurality of user-level threads of execution.

42. (Currently Amended) The computer readable medium ~~article of manufacture~~ of claim 35, wherein the computer readable medium ~~machine-accessible medium~~ further includes data that causes the machine to perform operations comprising sharing a system state among the plurality of user-level threads of execution.

43. (Currently Amended) The computer readable medium ~~article of manufacture~~ of claim 35, wherein the computer readable medium ~~machine-accessible medium~~ further includes data that causes the machine to perform operations comprising communicating between the plurality of user-level threads of execution via one or more shared registers.

44. (Currently Amended) The computer readable medium ~~article of manufacture~~ of claim 35, wherein an application program controls the plurality of user-level threads of execution directly, including scheduling of the plurality of user-level threads of execution, and wherein an operating system executed by the machine schedules the OS-generated thread.

45. (Currently Amended) The computer readable medium ~~article of manufacture~~ of claim 35, wherein the computer readable medium ~~machine-accessible medium~~ further includes data that causes the machine to perform operations comprising: associating the plurality of user-level threads of execution with the OS-generated thread; and suspending the plurality of user-level threads of execution belonging to the as generated thread when a context switch request is received through a one of the plurality of threads of execution.

46. (Currently Amended) The computer readable medium ~~article of manufacture~~ of claim 45, wherein the computer readable medium ~~machine-accessible medium~~ further includes data that causes the machine to perform operations comprising: storing one or more threads of execution states associated with the plurality of user-level threads of execution when the context switch request is received.

47. (Currently Amended) The computer readable medium ~~article of manufacture~~ of claim 35, wherein the computer readable medium ~~machine-accessible medium~~ further includes data that causes the machine to perform operations comprising: reporting one or more exceptions to a first user-level thread of the plurality of user-level threads of execution.

48. (Currently Amended) The computer readable medium ~~article of manufacture~~ of claim 47, wherein the computer readable medium ~~machine-accessible medium~~ further includes data that causes the machine to perform operations comprising: reporting the one or more exceptions from an application program; and determining whether to report the one or more exceptions to an operating system.

49. (Currently Amended) The computer readable medium ~~article of manufacture~~ of claim 48, wherein the computer readable medium ~~machine-accessible medium~~ further includes data that causes the machine to perform operations comprising prioritized-reporting of the one or more exceptions to the operating system; comprising

receiving the one or more exceptions concurrently via different user-level threads of the plurality of user-level threads of execution shreds; and

servicing one of the one or more exceptions according to the prioritized-reporting while suspending exception processing of other exceptions of the one or more exceptions.

50. (Currently Amended) The computer readable medium ~~article of manufacture~~ of claim 35, wherein the plurality of user-level threads of execution perform input/output (I/O) functions and computation functions.

51. (Currently Amended) A system, comprising:

a microprocessor to implement an instruction set architecture (ISA), the microprocessor to execute multiple operating system (OS)-generated threads, wherein the microprocessor is also to concurrently execute multiple shared resource threads

(shreds) associated with an OS-generated thread of the multiple OS-generated threads, each of the multiple shreds to be associated with a private state within the OS-generated thread and a shared state of the OS-generated thread, wherein each of the shreds is to be created utilizing hardware of the microprocessor in response to application program instructions of the ISA, and wherein a register that is part of the shared state of the OS-generated thread is adapted to be directly updateable by the multiple shreds to allow communication between the multiple shreds, wherein the register is also adapted to not be accessible by shreds that are not associated with the shared state of the OS-generated thread; and

a memory to store one or more instructions of the ISA that is to allow user-level multithreading operations.

52.–54. (Cancelled)

55. (Currently Amended) A system, comprising:

a microprocessor, including a plurality of user-level multithreading registers, wherein the registers are addressable by one or more application program instructions from a plurality of user-level threads to support communication among the plurality of user-level threads; and

memory coupled to the microprocessor, the memory to store the one or more application program instructions;

wherein the microprocessor is further to execute the plurality of user-level threads concurrently.

56. (Original) The system of claim 55, wherein the plurality of user-level multithreading registers further comprises a plurality of shared shred registers to facilitate communication between a plurality of shreds and to facilitate synchronization between the plurality of shreds.

57. (Original) The system of claim 56, wherein the plurality of user-level multithreading registers further comprises a plurality of shred control registers to manage the plurality of shreds.

58. (Previously Presented) The system of claim 57, wherein the microprocessor:
receives programming instructions to execute one or more shreds in accordance with the ISA;
configures one or more instruction sequencers via the ISA; and
executes the one or more shreds concurrently.

59. (Previously Presented) The apparatus of claim 32, wherein: the user-level exception mechanism is further to vector to a fixed location in order to allow the shred to service to the exception.

60. (Previously Presented) The apparatus of claim 32, wherein:
the plurality of instructions further include a system call instruction to explicitly invoke an operating system to service to the exception.

61. (Previously Presented) The apparatus of claim 34, wherein said prioritizer employs a round-robin scheme.

62. (Previously Presented) The article of manufacture of claim 35, wherein the one or more user-level programming instructions include an instruction to destroy one or more of the plurality of shreds.

63. (Previously Presented) The system of claim 51, wherein the one or more instructions include an instruction to create a shred without intervention of an operating system.

64. (Previously Presented) The system of claim 51, wherein the one or more instructions include an instruction to destroy a shred without intervention of an operating system.

65. (Previously Presented) The system of claim 51, wherein:
the user-level multi-threading operations include concurrent execution of two or more shreds associated with the same thread.

66. (Previously Presented) The system of Claim 55, wherein the memory is from a plurality of memory devices including DRAM, flash, and EEPROM.

67. (Previously Presented) An apparatus comprising:
a processor capable of user-level multithreading including:
a first group of resources to hold a per shared resource thread ("shred") application state for a first shred to be created by a first user-level instruction;
a second group of resources, which is to include a replicate of the first group of resources, to hold a per shred application state for a second shred to be created by a second user-level instruction; and
a third group of resources to be shared by the first shred and the second shred to hold a shared application state, wherein at least a portion of the third group of resources is to be directly updateable by the first and second shred for communication between the first shred and the second shred;
wherein the first, second, and third group of resources are private from another privileged-level software entity created thread; and
execution resources to concurrently execute the first shred and the second shred.

68. (Previously Presented) The apparatus of Claim 67, wherein: the first group of resources, the second group of resources, and the third group of resources include registers, and wherein the first user-level instruction and the second user-level instruction are an Instruction Set Architecture instructions to create a shred, which are to be associated with a first user software entity and a second user software entity, respectively.

69. (Previously Presented) The apparatus of Claim 68, further comprising a hardware re-namer to allocate the first group of registers as private registers and the third group of registers as shared registers based on a bit vector.

70. (Previously Presented) The apparatus of Claim 67, wherein the shared application state is to be shared by a privileged-level software entity thread and is not to be shared with other privileged-level software entity threads, and wherein the privileged level software entity is an operating system (OS).

71. (Previously Presented) The apparatus of Claim 67, wherein the second group of resources is a copy of the first group of resources, and wherein the first group of resources includes a combination of resources selected from a group consisting of general registers, floating point registers, SSE registers, instruction pointer, and flags, and wherein the third group of resources includes a combination of resources selected from a group consisting of general registers, floating point registers, shared communication registers, flags, memory management registers, address translation tables, privilege levels, and control registers.